Design of Synchronous Sequential Circuits

Objectives

1. Design of synchronous sequential circuits with an example.
2. Construction of state diagrams and state tables/
3. Translation of State transition table into excitation table.
4. Logic diagram construction of a synchronous sequential circuit

Sequential Circuit Design Steps

- The design of sequential circuit starts with verbal specifications of the problem (See Figure 1).

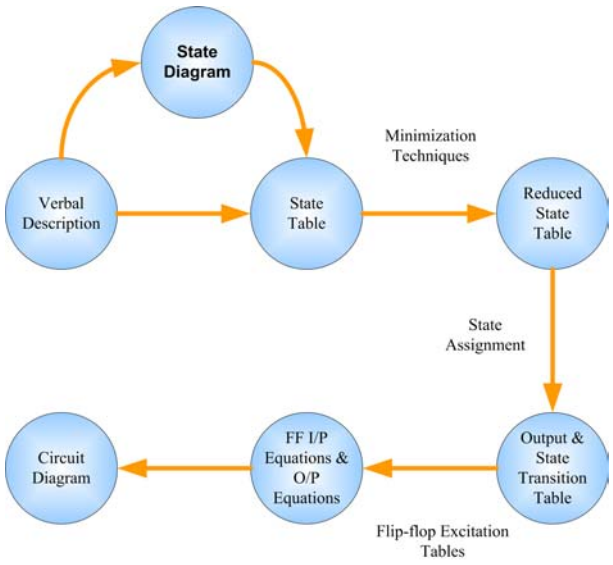

Figure 1: Sequential Circuit Design Steps

- The next step is to derive the state table of the sequential circuit. A state table represents the verbal specifications in a tabular form.
- In certain cases state table can be derived directly from verbal description of the problem.
- In other cases, it is easier to first obtain a state diagram from the verbal description and then obtain the state table from the state diagram.
- A state diagram is a graphical representation of the sequential circuit.
- In the next step, we proceed by simplifying the state table by minimizing the number of states and obtain a reduced state table.

- The states in the reduced state table are then assigned binary-codes. The resulting table is called output and state transition table.

- From the state transition table and using flip-flop's excitation tables, flip-flops input equations are derived. Furthermore, the output equations can readily be derived as well.

- Finally, the logic diagram of the sequential circuit is constructed.

- An example will be used to illustrate all these concepts.

Sequence Recognizer

- A sequence recognizer is to be designed to detect an input sequence of '1011'. The sequence recognizer outputs a '1' on the detection of this input sequence. The sequential circuit is to be designed using JK and D type flip-flops.

- A sample input/output trace for the sequence detector is shown in Table 1.

Table 1: Sample Input/Output Trace

Input	0	1	1	0	1	0	1	1	0	1	1	1	0	1	0	1	1	1	0	0
Output	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0

- We will begin solving the problem by first forming a state diagram from the verbal description.

- A state diagram consists of circles (which represent the states) and directed arcs that connect the circles and represent the transitions between states.

- In a state diagram:

    1. The number of circles is equal to the number of states. Every state is given a label (or a binary encoding) written inside the corresponding circle.

    2. The number of arcs leaving any circle is 2^n, where n is the number of inputs of the sequential circuit.

    3. The label of each arc has the notation x/y, where x is the input vector that causes the state transition, and y is the value of the output during that present state.

    4. An arc may leave a state and end up in the same or any other state.

- Before we begin our design, the following should be noted.

    1. We do not have an idea about how many states the machine will have.

    2. The states are used to "remember" something about the history of past inputs. For the sequence 1011, in order to be able to produce the output value 1 when the final 1 in the sequence is received, the circuit must be in a state that "remembers" that the previous three inputs were 101.

    3. There can be more than one possible state machine with the same behavior.

Deriving the State Diagram

- Let us begin with an initial state (since a state machine must have at least one state) and denote it with ***'S0'*** as shown in Figure 2 (a).

- Two arcs leave state ***'S0'*** depending on the input (being a 0 or a 1). If the input is a 0, then we return back to the same state. If the input is a 1, then we have to remember it (recall that we are trying to detect a sequence of 1011). We remember that the last input was a one by changing the state of the machine to a new state, say ***'S1'***. This is illustrated in Figure 2 (b).

- ***'S1'*** represents a state when the last single bit of the sequence was one. Outputs for both transitions are zero, since we have not detected what we are looking for.

- Again in state ***'S1',*** we have two outgoing arcs. If the input is a 1, then we return to the same state and if the input is a 0, then we have to remember it (second number in the sequence). We can do so by transiting to a new state, say ***'S2'***. This is illustrated in Figure 2 (c).

- Note that if the input applied is '1', the next state is still ***'S1'*** and not the initial state ***'S0'***. This is because we take this input 1 as the first digit of new sequence. The output still remains 0 as we have not detected the sequence yet.

- State ***'S2'*** represents detection of '10' as the last two bits of the sequence. If now the input is a '1', we have detected the third bit in our sequence and need to remember it. We remember it by transiting to a new state, say ***'S3'*** as shown in Figure 2 (d). If the input is '0' in state ***'S2'*** then it breaks the sequence and we need to start all over again. This is achieved by transiting to initial state ***'S0'***. The outputs are still 0.

- In state ***'S3'***, we have detected input sequence '101'. Another input 1 completes our detection sequence as shown in Figure 2 (e). This is signaled by an output 1. However we transit to state ***'S1'*** instead of ***'S0'*** since this input 1 can be counted as first 1 of a new sequence. Application of input 0 to state ***'S3'*** means an input sequence of 1010. This implies the last two bits in the sequence were 10 and we transit to a state that remembers this input sequence, i.e. state ***'S2'***. Output remains as zero.

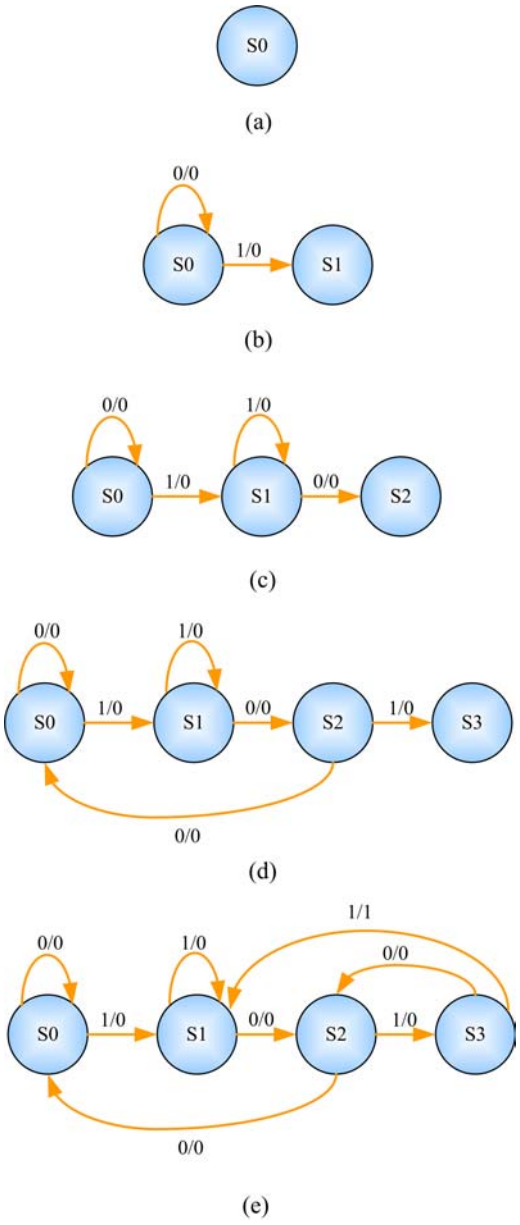

Figure 2: Deriving the State Diagram of the Sequence Recognizer

Deriving the State Table

- A state table represents time sequence of inputs, outputs, and states in a tabular form. The state table for the previous state diagram is shown in Table 2.
- The state table can also be represented in an alternate form as shown in Table 3.
- Here the present state and inputs are tabulated as inputs to the combinational circuit. For every combination of present state and input, next state column is filled from the state table.
- The number of flip-flops required is equal to $\lceil log_2(number\ of\ states) \rceil$.

- Thus, the state machine given in the figure will require two flip-flops $\lceil log_2(4) \rceil = 2$. We assign letters A and B to them.

Table 2: State Table of the Sequence Recognizer

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
S0	*S0*	*S1*	0	0
S1	*S2*	*S1*	0	0
S2	*S0*	*S3*	0	0
S3	*S2*	*S1*	0	1

Table 3: Alternative Format of Table 2

Inputs of Combinational Circuit		Next State	Output
Present State	Input		
S0	0	*S0*	0
S0	1	*S1*	0
S1	0	*S2*	0
S1	1	*S1*	0
S2	0	*S0*	0
S2	1	*S3*	0
S3	0	*S2*	0
S3	1	*S1*	1

State Assignment

- The states in the constructed state diagram have been assigned symbolic names rather than binary codes.
- It is necessary to replace these symbolic names with binary codes in order to proceed with the design.
- In general, if there are ***m*** states, then the codes must contain ***n*** bits, where $\mathbf{2}^{n} \geq \boldsymbol{m}$, and each state must be assigned a unique code.
- There can be many possible assignments for our state machine. One possible assignment is show in Table 4.

Table 4: State Assignment

State	*Assignment*
S0	00
S1	01
S2	10
S3	11

- The assignment of state codes to states results in state transition table as shown.

- It is important to mention here that the binary code of the present state at a given time ***t*** represents the values stored in the flip-flops; and the next-state represents the values of the flip-flops one clock period later, at time ***t+1***.

Table 5: State Transition Table

Inputs of Combinational Circuit		Next State	Output
Present State	Input		
A B	***X***	***A B***	***Y***
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	1 0	0
0 1	1	0 1	0
1 0	0	0 0	0
1 0	1	1 1	0
1 1	0	1 0	0
1 1	1	0 1	1

General Structure of Sequence Recognizer

- The specifications required using JK and D type flip-flops.

- Referring to the general structure of sequential circuit shown in Figure 3, our synthesized circuit will look like that as shown in the figure. Observe the feedback paths.

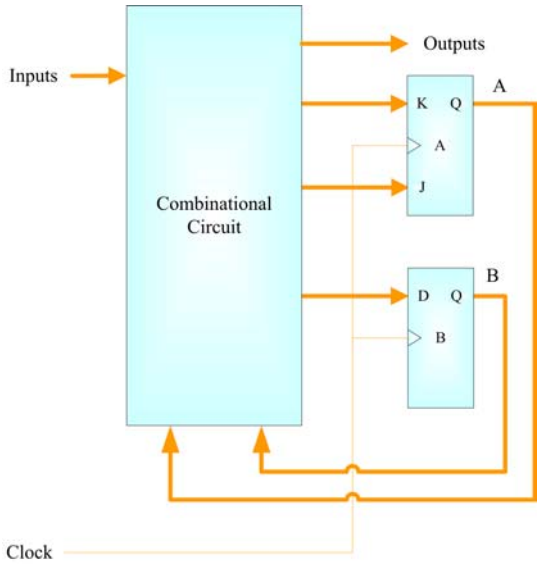

Figure 3: General Structure of the Sequenc Recognizer

- What remains to be determined is the combinational circuit which specifies the external outputs and the flip-flop inputs.

- The state transition table as shown can now be expanded to construct the excitation table for the circuit.

- Since we are designing the sequential circuit using JK and D type flip-flops, we need to correlate the required transitions in state transition table with the excitation tables of JK and D type-flip-flops.

- The functionality of the required combinational logic is encapsulated in the excitation table. Thus, the excitation table is next simplified using map or other simplification methods to yield Boolean expressions for inputs of the used flip-flops as well as the circuit outputs.

Deriving the Excitation Table

- The excitation table (See Table 6) describes the behavior of the combinational portion of sequential circuit.

Table 6: Excitation Table of the Sequence Recognizer

Present State		***Input***				***Flip-flops Inputs***		
A	***B***	***X***	***A***	***B***	***Y***	J_A	K_A	D_B
0	0	0	0	0	0	0	X	0
0	0	1	0	1	0	0	X	1
0	1	0	1	0	0	1	X	0
0	1	1	0	1	0	0	X	1
1	0	0	0	0	0	X	1	0
1	0	1	1	1	0	X	0	1
1	1	0	1	0	0	X	0	0
1	1	1	0	1	1	X	1	1

- For deriving the actual circuitry for the combinational circuit, we need to simplify the excitation table in a similar way we used to simplify truth tables for purely combinational circuits.

- Whereas in combinational circuits, our concern were only circuit outputs; in sequential circuits, the combinational circuitry is also feeding the flip-flops inputs. Thus, we need to simplify the excitation table for both outputs as well as flip-flops inputs.

- We can simplify flip-flop inputs and output using K-maps as shown in Figure 4.

- Finally the logic diagram of the sequential circuit can be made as shown in Figure 5.

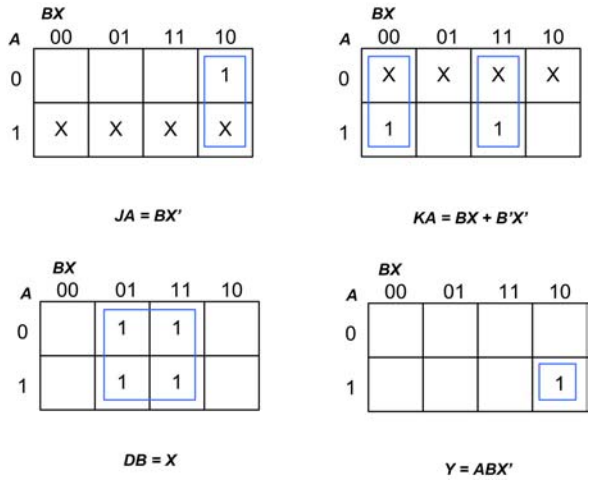

Figure 4: Input Equations of the Sequence Recognizer

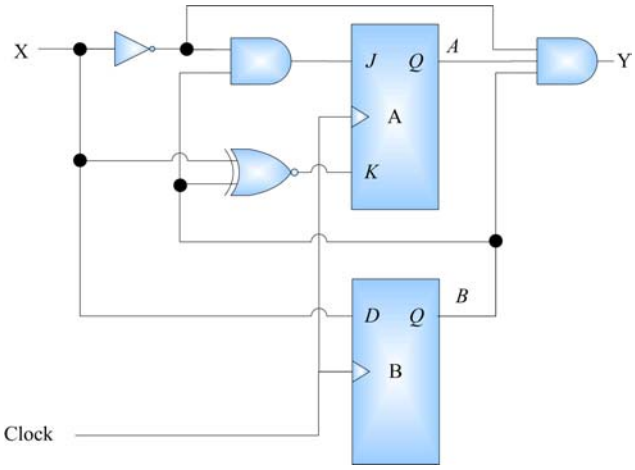

Figure 5: Circuit Diagram of the Sequence Recognizer